Monday November 8, 2004
Meeting Log

Purpose: Read XMPP IM
Time: November 14, 2004 Sunday. 1:00 pm
Location: Sun Lab
Secretary: Brandi Soggs
Present: Brandi, Jason, Jonas, Steven, Matt

Items Discussed:
Weekly report-

Matt, 1 hour
Jason, 2 hours
Steven, 2 hours (doc), 30 minutes (webpage)
Jonas, 1 hour
Brandi, 2 hours

Overview-

XMPP (Extensible Messaging and Presence Protocol) is a protocol for streaming XML elements in order to exchange messages and presence information in close to real time
If we don't have to worry about presence, we don't have to do it
We're only worrying about jabber client namespace

Message Stanza <message

Type attribute:
- chat (one to one chat, should maintain a history)
- error (don't send errors in response to errors)
- groupchat (maintain a history)
- headline (automated stuff, like AOL stock ticker, no response needed)
- normal (other, no chat history… inkboard?)

Type is optional, but recommended. Generally you want to reply to one type with the same type (chat reply is type chat), except for errors
Child elements
- <subject/> (no attributes)
- <body/> (no attributes)
- <thread/> (unique id for each conversation, copied back and forth with each reply)

Presence <presence

Type is optional. No type tells the server that the user is online and available for chat
Types:
- unavailable
- subscribe (a user wants to add this user to buddy list)

- unsubscribe (delete from buddy list)
- subscribed (on buddy list)
- unsubscribed (not on buddy list)
- probe (is user online?)
- error

Child elements:

- <show/> (away, chat, do not disturb… no attributes)
- <status/> ("In a meeting")
- <priority/> (small number)

MUST NOT contain more than one <show/> element

IQ <iq

Whatever you want the iq to do is handled in the attributes

<session …>

Look to the previous notes

Session Establishment

We can extend the namespace all we want. We can count on the server to route the information, and other clients will ignore it

A client must complete stream authentication before attempting to establish a session

"Session" is not defined, but we think it means back and forth messages

If a server supports sessions, it will advertise it with a session element 'urn:ietf:params:xml:ns:xmpp-session'

Not going to resolve this "security? y/n" issue any time soon just by reading this, keep getting mixed signals

Exchanging Messages

Recipient: provide the JID in the to: attribute of the <message/> stanza

Resource is not required in the first message to the recipient, but after that it should be included (to: JID@domain/resource)

Not certain what to put in the xml:lang, but we don't have to worry about that

Exchanging Presence Information

Presence information is specified by type (no presence = available)

When first signing on (initial presence)-

- <presence (no to) (no type)
- Sends presence probes to each user on buddy list
- Sends broadcast to each user subscribed to the signing on user of the initial presence (with full JID@domain/resource)

Presence probes: a bunch of clients are connected to a server, to each of whom another client is subscribed to. The user signs on, and the server sends presence probes to the subscribed-to users to ask for their state (builds the user's buddy list).

There's a state for blocking traffic

We're not going to use the subscription for inkboard clients

Lots of examples on page 20-25

Managing Subscriptions

They seem like iqs that don't get responses

We have to be subscribed to inkboard clients to get all the information. However, we don't want to have to implement add buddy/delete buddy within inkboard, and managing a special inkboard JID's subscriptions via external client seems rather annoying

We think it might be possible to just have one JID for external chat client and for inkboard. They will ignore each other's messages. Problem: will logging off one client log off the other client, too? Will presence messages reach both inkboard and the external chat client?

It's looking like we will have to have two separate JIDs, one for inkboard and one for external client.

- hidden buddy list- everyone we ever talk to just gets put on the list (potentially very large)
- Visible buddy list- a lot of managing expectations come with showing a buddy list that we don't want to have to handle
- All subscriptions are handled via an external client (annoying to the user). Probably going to have to go with this option

Compliancy-

We aren't going to be a compliant jabber client.

- we're going to handle IM-specific semantics of XML stanzas
- We're probably NOT going to do the subscription stuff
- We're NOT going to have end to end encryption

How to test this?

Generally use a public jabber server

Perhaps we can get a server implementation and put it on a computer here?

Will we be able to make much out of the server errors?

What if we had a packet sniffer check for errors?

Questions

- Do we have to implement security?
- How are we going to handle the buddy list?
- Look at JEP-0045 for groupchat? Potentially Jonas will run off and read this document, tell us if we need to know more. Or, we'll just try to make the most basic group chat possible and see if we need to research it more if that fails miserably.

TO DO:

Send Brandi the times when you have commitments for next quarter (work, clubs, etc) other than your class schedule. Brandi will compile a list of mutually available times in an effort to meet during the day